Project 13: Software-based Approach for Real-time Photoacoustic Imaging using Vendor Independent US Beamformed Data

Howard Huang

Mentors: Haichong "Kai" Zhang, Emad Boctor

May 27, 2016

Medical UltraSound Imaging and Intervention Collaboration (MUSiiC) Lab, Johns Hopkins University, Baltimore, MD

Table of Contents

I Background

Photoacoustic (PA) imaging is a medical imaging modality based on the photoacoustic effect, where ultrasound signals are generated from an object due to absorption of optical energy [1, 5]. The effect is commonly achieved through exposure to laser pulses [5]. Given that different tissues and objects have varying degrees of optical absorption, PA imaging has shown the potential for cancer detection [5], blood vessel visualization [6], and instrument tracking (i.e. the tracking of radioactive seeds relative to the prostate in brachytherapy) [4]. Unfortunately, current research and usage of PA imaging are constrained by additional hardware costs, due to the fact that PA signals are incorrectly processed by existing Ultrasound (US) systems. This problem arises from the fact that US systems beamform the PA signals as echoes from the US transducer instead of directly from illuminated sources, resulting in defocused images [6]. As a result, current implementations of PA imaging include reliance on parallel channel acquisition from research US platforms, which are not only slow (as channel data can be in the range of gigabytes per image frame) and expensive [4], but are also not approved by the FDA for clinical use. Other methods include modifying the US beamformer to better focus PA signals [2], but these methods are limited in that they are vendor-dependent, as each US system will have a different beamformer with some hardwired or impossible to alter to the desired setting. Given that real-time access to beamformed data is available on many current clinical US systems, the MUSiiC lab has developed a re-beamforming algorithm that will refocus this data into a clear image for PA signals [6]. Therefore, the goal of this project is to implement a real-time version of this algorithm and to demonstrate its performance on a conventional US platform (the Ultrasonix SonixTouch US imaging machine).

II Problem

As mentioned previously, US systems cannot image PA signals due to incorrect beamforming. As depicted in the following figure, PA signals originate from an illuminated source, whereas US signals are generated by the US probe and reflected back by objects with different acoustic impedances:

Figure 1: Differences in US vs PA imaging.

Observe that US signals are double trip signals (travel from probe to object to probe) whereas PA signals are single trip (from objects to probe), resulting in differences in their time delays. The result is that PA signals have a travel time of $t = d/c$ and US signals have a travel time of $t = 2d/c$, where d is the distance travelled and *c* is the speed of the sound wave [6]. As a result, different beamformers are required to focus US and PA signals into an image. A US beamformer applied onto PA signals will generate defocused signals at an incorrect depth as demonstrated by the figure to the left displaying a B-mode image of US beamformed data from a point source signal. Imaged properly, the signal should have appeared as a single bright dot.

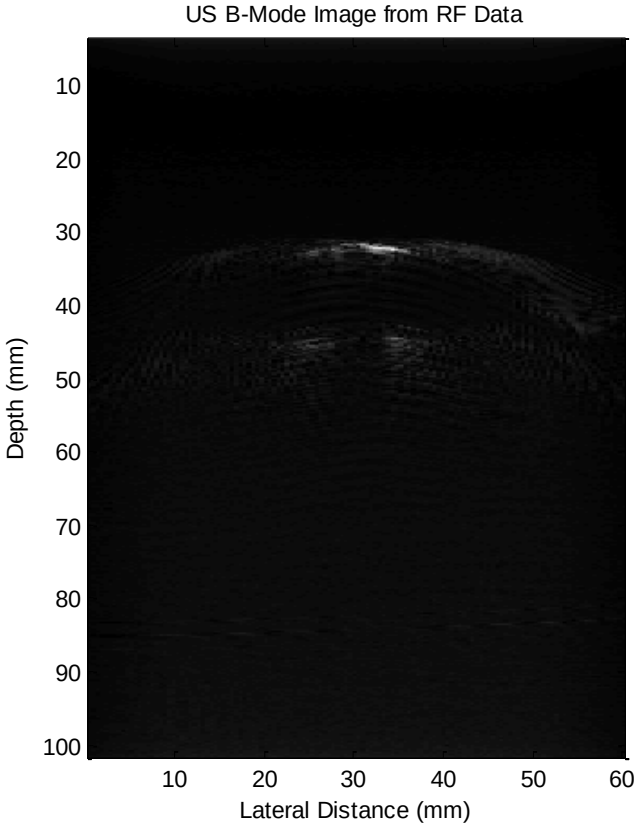


Figure 2: Example of unfocused PA image (poor spatial resolution).

IIIa Technical Approach – PA Imaging Workflow and Algorithm

Since the US beamformed data is defocused but still contains the PA signal captured by the US probe, our approach is to develop software to reprocess this data into a PA image instead of processing prebeamformed channel data (which is not accessible on most clinical US systems). The advantages to this approach is that US beamformed data exists in the imaging workflow of any US system regardless of platform or vendor, with many systems having real-time access to this data. Thus, by ensuring that our algorithm can rapidly refocus US beamformed PA data, we can create a real-time PA imaging workflow for any conventional US system assuming real-time access to RF (radio frequency) beamformed data from such systems, as depicted below:

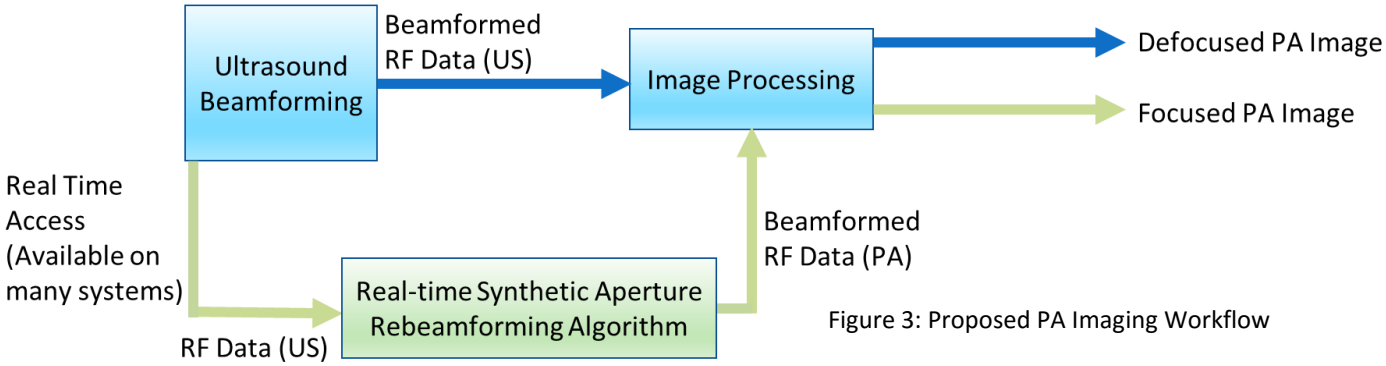


Figure 3: Proposed PA Imaging Workflow

The refocusing algorithm uses synthetic aperture rebeamforming, an approach developed by the MUSiiC lab to sequentially refocus PA signals at varying depths. Since PA signals were beamformed with a time delay that is off by a factor of 2, the depth of each PA signal is roughly halved in the beamformed data. As a result, the beamformed data can actually be reinterpreted as raw signal data, with the depths of each signal dived by two. The travel distance r of the signal is now $r = \sqrt{\left(\frac{Y_n}{2}\right)^2 + X_n^2}$ where Y_n and X_n are respectively the depth and lateral distance of the signal source from transducer element. The time of flight t of the signal is thus

$t = r/c$ where c is the speed of sound. Applying another round of beamforming using these corrected time delays will result in a refocused PA image. Rescaling the image depth by a factor of 2 will then correct the PA signal's depth.

IIIb Technical Approach – PA Algorithm Validation (Simulation & Demo)

Our first goal was to develop a fast implementation of the synthetic aperture rebeamformer. This algorithm is implemented in C++, with documentation on its installation and usage on the CIS webpage (project 13). The implementation was then verified using simulated US beamformed data (simulation code and datasets available on the wiki as well). The simulated data contains multiple defocused PA signals that should appear as distinct point sources when properly rebeamformed.

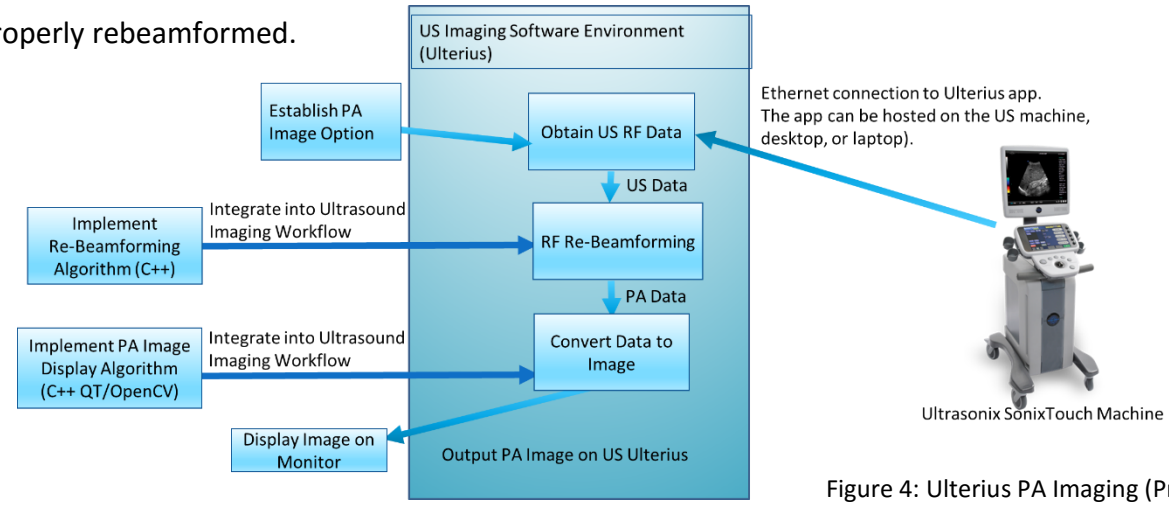


Figure 4: Ulterius PA Imaging (Proposed modifications to workflow).

After verification, we proceeded to integrate our algorithm into a real-time imaging workflow (diagram for workflow shown above). We built and modified the Ultrasonix Ulterius application, which allows for real-time acquisition of B-mode and RF data from Ultrasonix machines, to include a PA-imaging option. When configured to this option, the system will automatically rebeamform RF signals received from the Ultrasonix SonixTouch machine. The system then will

apply scan-conversion onto the data to allow it to be displayed on screen within the Ulterius application. We included a data collection feature on the Ulterius interface in order to collect real data to calibrate our rebeamformer algorithm. To assess the real-time performance of our algorithm we also included a frame rate output option, which calculates the time delay between images and outputs the information to the Ulterius terminal window.

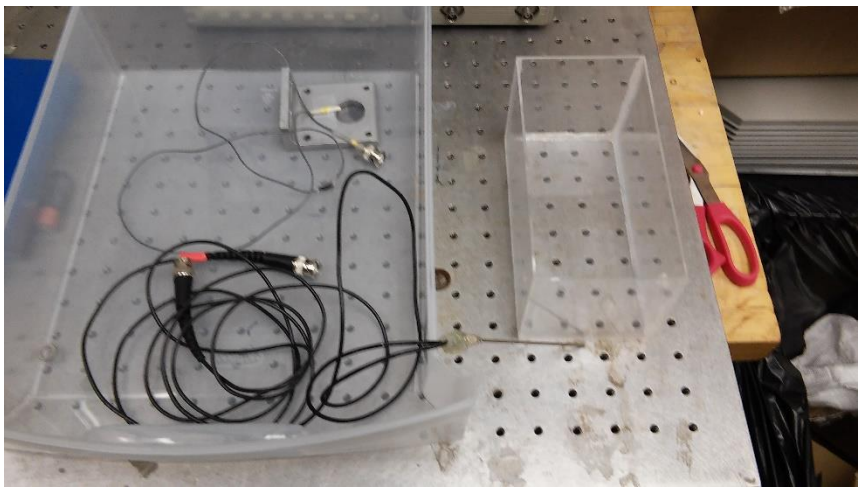


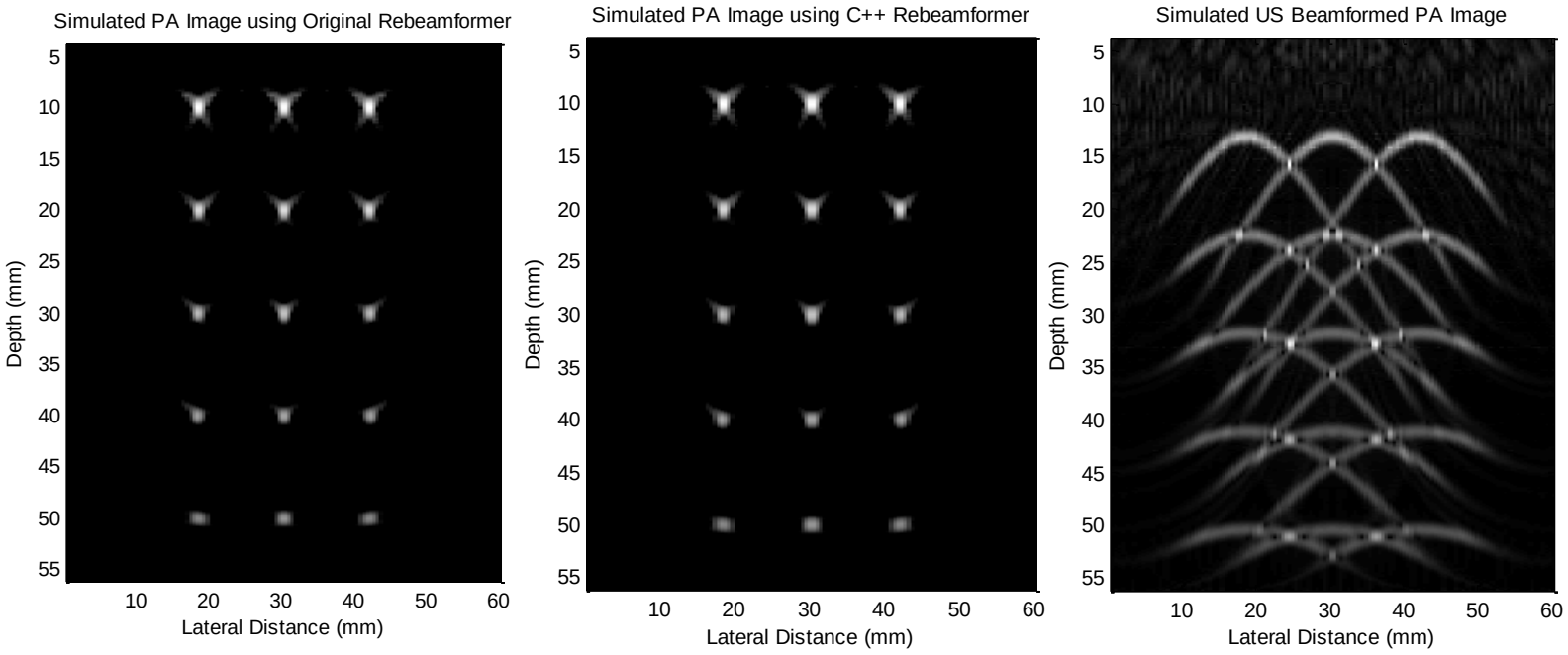
Figure 5: Pseudo-PA Imaging Setup (Using water chamber with PZT Element).

Our current experimental setup uses a PZT element to emit ultrasound signals within a chamber of water. This simulates the effect of a PA generated signal while also allowing for an open display of our PA imaging demo as no high energy lasers are used.

Future work includes testing this system on real PA systems. To prepare for an upcoming PA demo (using an LED energy source), we have also implemented an image averaging option into our Ulterius interface. This option is useful for the LED setup, because the LED system outputs a low level of optical energy that is diffused across the medium (yielding more noise than a traditional PA laser system). The averaged signal should reduce the level of noise encountered when testing this future system.

IVa Results – Simulated PA Signal and Real PZT Signal

Below are the results before and after application of the PA rebeamformer on simulated and real pseudo-PA signals. For the simulated data sets, we compared our C++ implementation to the MUSiiC's verified (but slower) version of the synthetic aperture rebeamformer.



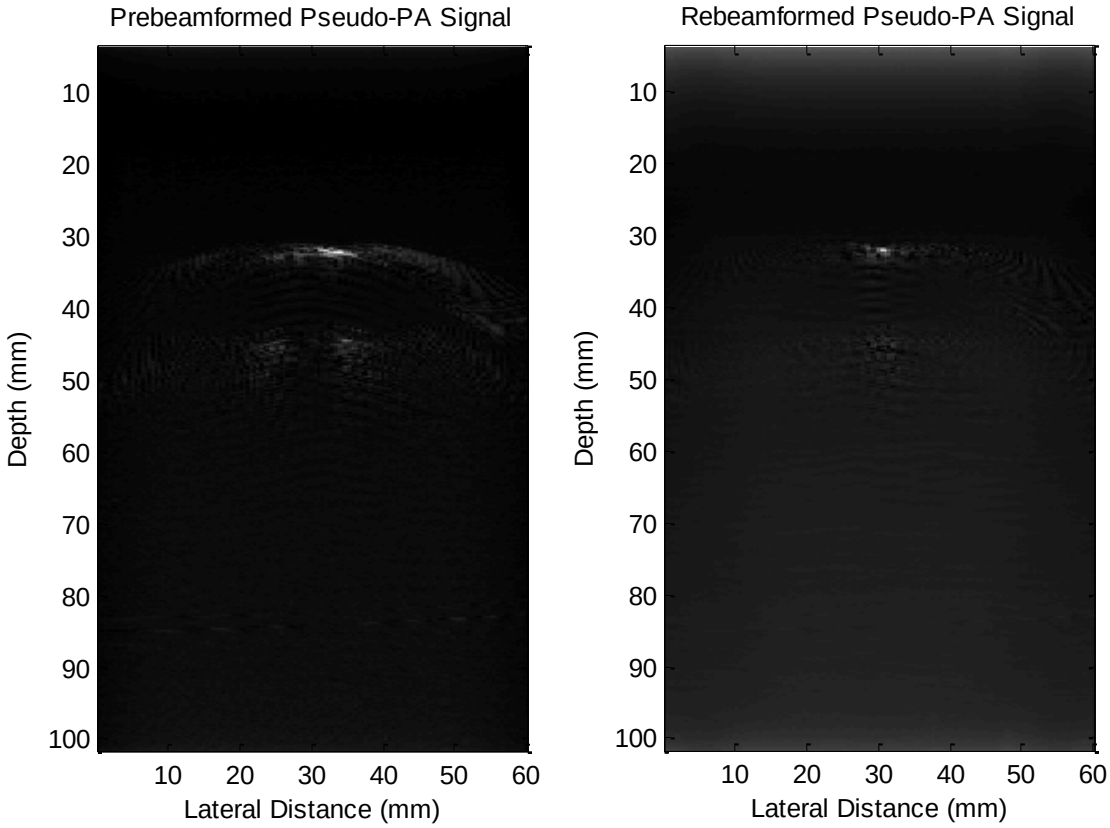


Figure 6: Results of Real and Simulated PA Imaging Experiments (With and without PA Rebeamformer).

From the simulated dataset, it is clear that our C++ implementation functions as intended (full match with existing rebeamforming algorithm developed by the MUSiiC lab). Furthermore, for the pseudo-PA signals acquired from a real experimental setup, we can see significant lateral resolution improvements (from 10mm to 1 mm) in the location of the PA signal relative to the transducer. The same improvements can be seen in real-time for our Ulterius imaging demo (video results on wiki, image results on next page).

IVb Results – Ulterius Demo Interface and Performance

Below is our current modifications to the Ulterius interface. We added the PA imaging option within the acquisition checklist (Rf beamformed option) and also modified the interface to process and display the resulting data. We also created data output and image averaging options in anticipation of a future LED PA imaging demo.

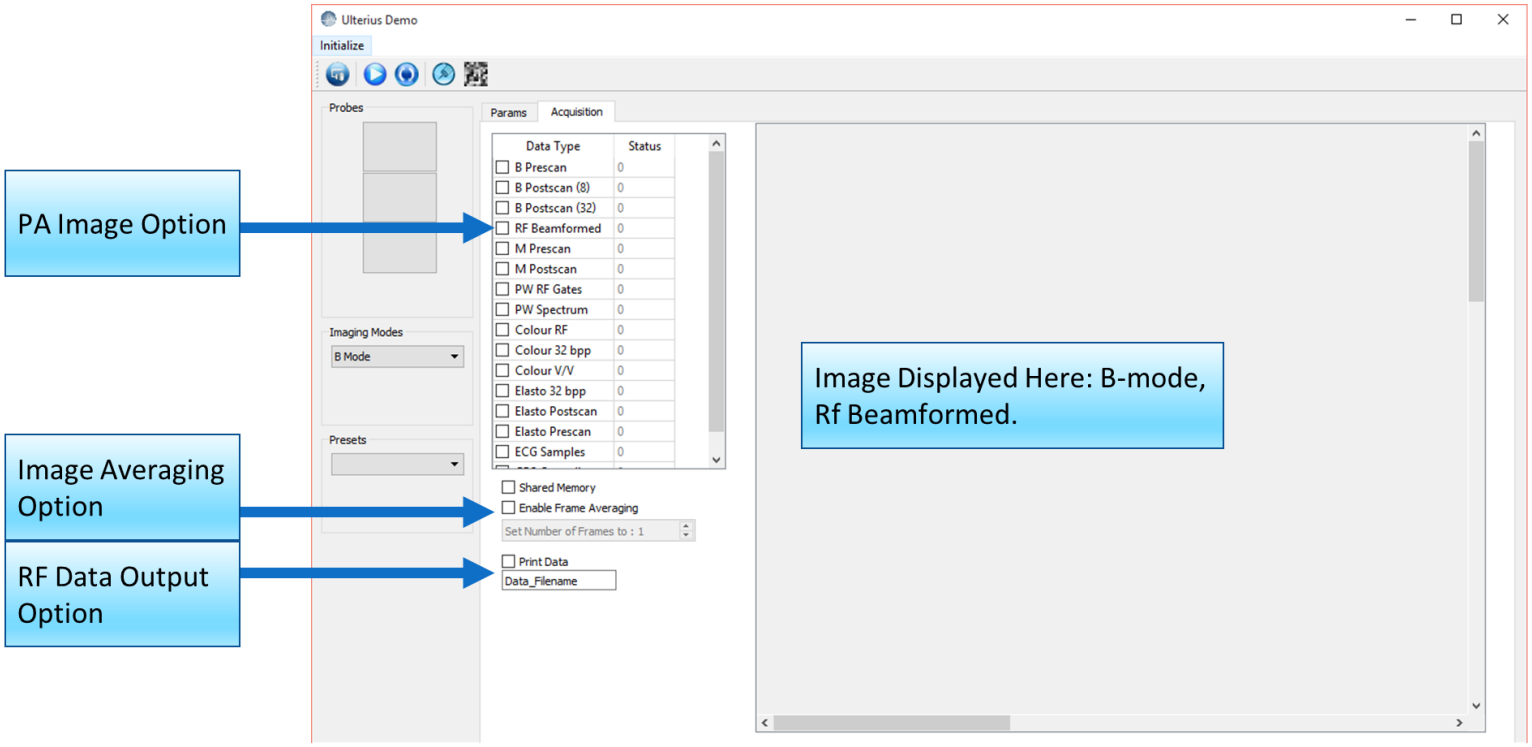


Figure 7: Modified Ulterius Interface and Features.

Examples of B-Mode and RF images captured on the interface (rescaled for comparison).

Significant improvement of resolution in the PA signal can be observed between the unfocused B-mode image (originally present on SonixTouch platform) and the new PA imaging mode:

B Mode Image: 6 cm width by ~2 cm depth

Rebeamformed PA Image: 6 cm width by ~4

Figure 8: Original US Image of PA Signal vs Rebeamformed PA Image.

Finally, we calculated the frame rate of our system by measuring the time delays between acquiring new image data (the system processes one image before proceeding to the next). As shown in the data below, frame rate is dependent on the image depth, due to higher depths requiring more samples per element to be processed by our algorithm:

We found that for a low depth (lowest number of samples per ultrasound element), the system can actually process up to 50 images per second. Even at a maximum depth (highest number of samples per element), the system can process up to 4 frames per second, which can still be considered real-time performance.

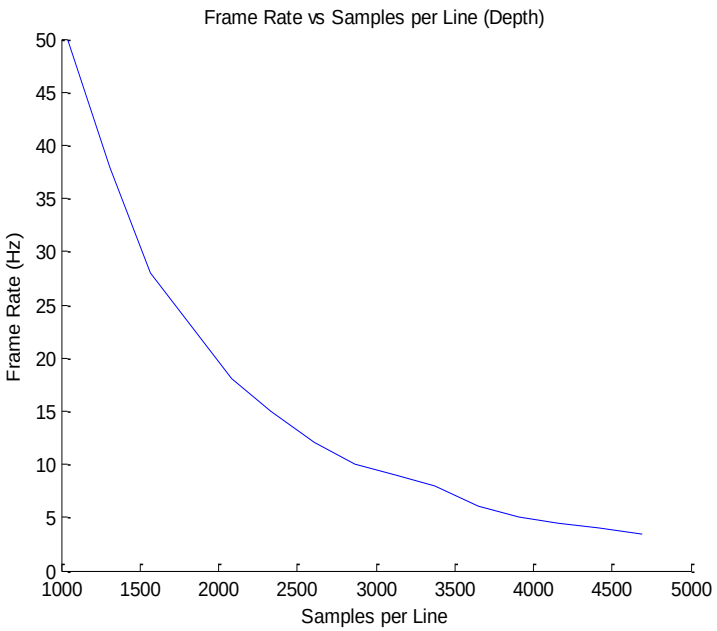


Figure 9: PA Imaging Frame Rate (Ulterius).

V Significance

As mentioned previously, past implementations of PA imaging on US systems were not only expensive, but also slower compared to our current implementation. A previous attempt on PA imaging using channel acquisition not only resulted in a maximum frame rate of 3-4 Hz, but could also only work for experimental US systems [3]. Our current approach has demonstrated the viability of PA imaging for almost any conventional US system with real-time RF data access with the current maximum framerate around 50 Hz. Given that our current algorithm only uses precomputed delay tables to increase performance, it is certainly possible to parallelize the rebeamforming process by creating multiple threads to rebeamform the data across separate lines/depths, and increase frame rates even further. If this approach is integrated into existing and future clinical US systems, we can expect to see increased research and usage of PA imaging for medical purposes at a fraction of the current costs as our imaging method contains no new hardware dependencies.

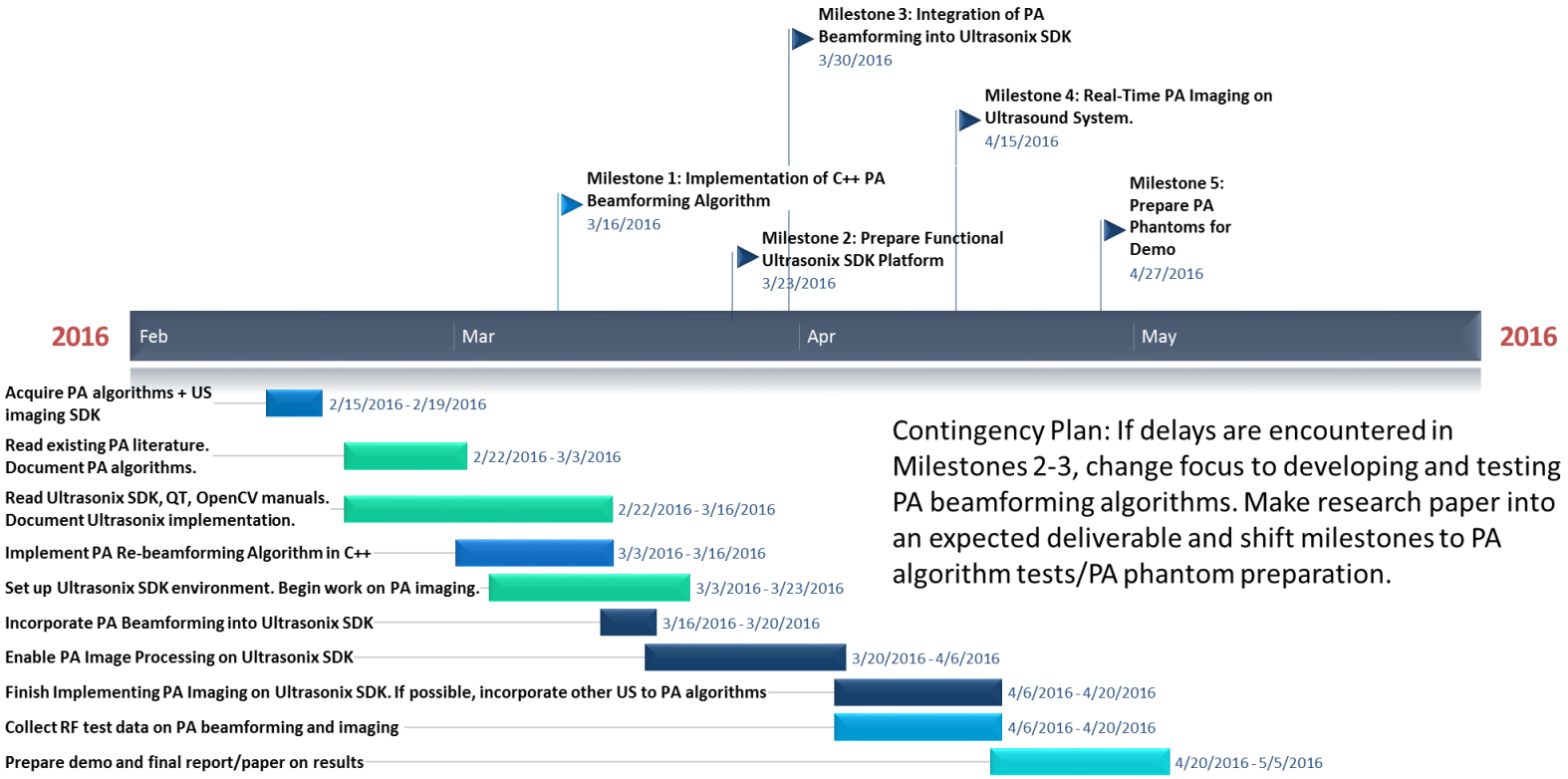
VI Management Summary

All work on the C++ rebeamformer and Ulterius demo presented in this report was achieved by Howard Huang under the mentorship of Kai Leng and Dr. Emad Boctor. The simulation US-PA signal data and code (used to initially verify C++ rebeamformer) were made by Kai during development of the synthetic aperture PA rebeamforming algorithm.

To increase rebeamforming speed, the C++ rebeamformer also uses precomputed delay tables and was calibrated to run in real time in the Ulterius demo. Due to differences in RF signal

formats (simulated data uses double type, actual RF data from the SonixTouch machine is uint_16 type), two versions of C++ rebeamformers (an initial version for simulated data, an optimized one for real RF data) are included on the Wiki.

Our initial plan in our project proposal for the course were:



Contingency Plan: If delays are encountered in Milestones 2-3, change focus to developing and testing PA beamforming algorithms. Make research paper into an expected deliverable and shift milestones to PA algorithm tests/PA phantom preparation.

During the semester, we encountered significant challenges in milestones 2 and 3. We realized during spring break that Ulterius app was only a data transfer demo (no imaging option, with dependencies like Visual Studios 2010 no longer publically available). The MUSiiC lab did have a reference version of Ulterius that had basic image display and data processing, but was built for a separate project (not calibrated for RF signals). As a result, project goals over spring break shifted to building the functional Ulterius build with imaging and understanding the RF data format. Fortunately, the student had anticipated difficulty in creating an Ulterius demo (past

research projects suggested generous schedules for these milestones) and the milestones 2-3 were completed around the beginning of April (within range of expected deadlines). Milestone 4 was the last challenge. Without precomputed delay tables and other code optimization, the initial rebeamformer only could produces images around 1 Hz, which was borderline real-time performance. Therefore, we decided to postpone one of our optional max deliverables in favor of algorithm optimization, which proved to be the correct decision (as framerates were eventually improved to 50 Hz).

Our planned deliverables were:

Minimum:

1. Documentation of PA re-beamforming algorithm and its integration into an US visualization platform. ✓
2. Implementation of C++ re-beamforming algorithm. ✓
3. Scripts to debug algorithm with simulation data. ✓

Expected:

1. US platform updated for real-time PA imaging.* Requires integrating our PA rebeamformer into system. ✓
2. Construction of PA/US phantoms/experiments to test PA imaging system. (PA phantoms available) ✓
3. Test results of PA imaging system using real RF US data. (Results in PA-RF data or image format) ✓

Maximum:

1. Implementation of additional PA image algorithms (inverse beamforming, US visual data conversion) in completed PA imaging system.
2. Summary of PA imaging system or real-time PA rebeamformer in a paper for submission. (This report comprises the base of our paper submission) ✓
3. An in-class live demo of real-time PA imaging system. ✓

We have accomplished all deliverables except the implementation of additional PA image algorithms (an optional max deliverable), since we were in favor of improving our current rebeamformer for real-time performance. Once the semester ends, we intend to publish this

report (possibly after the MUSiiC lab's publication of the rebeamformer algorithm) to raise awareness of our PA imaging method.

Future work includes acquiring more PA data (potentially from the LED demo on Friday, or from the incoming PA laser system over the summer) and integrating those results into this report for later submission. Further refinement of the PA rebeamformer (via parallelization) or implementation and comparison of alternative PA imaging methods (i.e. inverse beamformer) are also potential future steps.

Overall, I have increased my understanding of PA imaging, as well as application development (learning to use QT and OpenCV was a great but time-consuming experience), and improved planning/management skills from my time in this project.

VII Acknowledgements

I would like to thank my mentors Kai Zhang, Dr. Emad Boctor for providing excellent support and allowing me to work with them in this project as well as Chen Lei for providing an example of a modified Ulterius application for B-mode imaging (which I greatly appreciated and referenced while creating a version for PA imaging).

I would also like to thank Dr. Bell for teaching me the basics behind current medical imaging modalities which sparked my interested in PA imaging research projects as well as Dr. Russell Taylor and Alexis Cheng for being fantastic CIS instructors.

VIII Appendix (Documentation. Also Available on CIS II Wiki)

Live Demo Setup Instructions:

Download the compiled Ulterius demo package on wiki.

Setup the SonixTouch machine:

1. Set IP address to 192.168.112.4 and subnet mask to 255.255.255.0
2. On B-Geom: Set all line densities to 128 and accumulator to 1.
3. On B-TX: Set pulse index A to 3 (turns of transducer).
4. Connect laptop or desktop by Ethernet to US machine.
5. Configure laptop's IP to 192.168.112.1 and subnet mask to 255.255.255.0
6. Run the Ulterius Demo on the laptop.
7. Once the demo is started press the connect buttom on the upper left corner of the interface.
8. Press the color button on the Ulterius device followed by B-mode image button.
9. Press the RF-beamformed option on the Ulterius interface.
10. Real time images will now be displayed.
11. To save data, enter a file name into the data_filename input on the interface. Press the save data button to record one image frame of rf data.
12. If image averaging is desired, press the enable averaging button and input number of frames to be averaged in below input tab.

PA Simulation Demo Instructions:

1. Download the PA Simulation Code package.
2. Run the generate_US_Simulation.m file to run Kai's simulation + my C++ rebeamformer. (Note, majority of code here is from the MUSiiC lab, I used the code mainly to assess my C++ rebeamformer).
3. Results between the two rebeamformers will be shown at the end.

PA Real Data Analysis Instructions:

1. Download the PA Real Data Analysis Code package.
2. Run the Analyze_Data.m file to run the optimized C++ rebeamformer on real PA RF data acquired from the Ulterius demo.
3. Results are displayed at the end.

Overview of PA Rebeamformer Algorithm and modified Ulterius (packaged in sdk6 on wiki):

Function call: double* pa_rebeamforming(const int ns, const int nl, double channelSpacing, double speedSound, double freq, double* rf)

Function input description:

1. rf = array of RF data (in double or int_16 format)
2. ns,nl = dimensions of RF data(# samples per line, # lines)
3. channelSpacing = distance between lines
4. speedSound = speed of sound (for depth calculations)
5. freq – freq of sampling.

This algorithm calculates a new time delay and then applies another round of dynamic beamforming on data. The beamforming effectively sums together elements in the array based on this time delay (where depth is roughly 1/2 that of the original beam-forming).

Output = rebeamformed rf array (of same dimension as input array)

Note: to speed up the algorithm, we computed the delay tables separately in our real RF data version. The behavior and parameters of the algorithm are not changed but are reorganized, with the channelSpacing, speedSound, and freq parameters are relocated now to the precomputation code, which is executed once. The precomputation speeds up the performance of the system significantly.

Ultrasonix Ulterius Documentation and Installation: Ulterius source code is provided within the Ultrasonix SDK package (sdk607) on Wiki

To build Ulterius code on Windows:

1. Download Microsoft Visual Studios (2010 version suggested by Ultrasonix documentation, but difficult to find).
2. Download CMake 2.8.8, QT4 (4.8.7), and opencv (2.4.11).
3. Run CMake 2.8.8 on sdk607 directory (create a new directory for project files)
4. Establish QT4 and opencv libraries in Visual Studios 2010.

Implementation of Image Panel on Ulterius Interface: (Many thanks to Lei for providing a reference Ulterius build with QT visualization)

- Modified Ulterius.ui file. Added in a viewing plane object using QT software interface on VS 2010.
- Added new options for image averaging and data saving.

In UlteriusDemo.cpp:

- Modified onNewData function (recieves data as an array from the system) to transfer array to processFrame function.
- Added processFrame function - A new function built on the UlteriusDemo.cpp. As inputs: it takes in data (1D array of chars/ints) and data type (int). Data is then transformed into a 2d array of pixels (QTarray) and send into the viewing plane object.
- Adapted PA rebeamformer into Ulterius to allow for real-time PA imaging. Made modifications to allow for image averaging and scan conversion as well.
- Added a saving option and image averaging options (in the Ulterius interface).

Effectively, the workflow for the new Ulterius interface is: Data from system –> calls onNewData function –> calls processFrame function –> display image. By adding our rebeamforming function into onNewData function (effectively convert RF array data to B mode array data) we can then send the data and proper image type into the processFrame function to display a properly beamformed image for PA signals. We can also apply envelope detection and log compression to refine the image further.

IX References

1. Bell, Alexander Graham. "LXVIII. Upon the production of sound by radiant energy." The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 11.71 (1881): 510-528.

2. Harrison, Travis, and Roger J. Zemp. “The applicability of ultrasound dynamic receive beamformers to photoacoustic imaging.” Ultrasonics, Ferroelectrics, and Frequency Control, IEEE Transactions on 58.10 (2011): 2259-2263.

3. Kang, Hyun-Jae, et al. “Software framework of a real-time pre-beamformed RF data acquisition of an ultrasound research scanner.” SPIE Medical Imaging. International Society for Optics and Photonics, 2012.

4. Kuo, Nathanael, et al. “Real-time photoacoustic imaging of prostate brachytherapy seeds using a clinical ultrasound system.” Journal of biomedical optics 17.6 (2012): 0660051-0660057.

5. Mehrmohammadi M, Yoon SJ, Yeager D, Emelianov SY. Photoacoustic Imaging for Cancer Detection and Staging. *Current molecular imaging*. 2013;2(1):89-105. doi:10.2174/2211555211302010010.

6. Siphanto, Ronald I., et al. "Imaging of small vessels using photoacoustics: an in vivo study." *Lasers in surgery and medicine* 35.5 (2004): 354-362.

7. Zhang, Kai, et. al. “Synthetic Aperture Based Photoacoustic Image Re-beamforming From Ultrasound Post-beamformed RF Data”. Unpublished Manuscript (will be submitted for publication).